* Error, Fault & failure :

→ **Error :**

- Error refer to difference between actual o/p of s/w & correct output of s/w.

→ **Fault :**

- Fault is a condition that pause a system to fail in performing its required functions.

→ **Failure :**

- It is inability of a system or component to perform its required functions according to its specification.

* Testing Objective :

- There are some testing objective as follow.

A) Testing is a process of execution a program with intent of finding error.

B) A Good Test case is one that has high probability of finding undiscovered errors.

C) A successful test is one that uncover undiscovered errors.

* Testing Principle :

- Software engineer must understand basic principle that guides software testing.

1) All test should be traceable to customer requirenment.

- The objective of software testing is to uncover errors.

- It follow that most defect from customer point of view are those that causes the program to fail to meet its requirenment.

2) Test should be planned before testing begins.

- Test planning can begin as soon as analysis phase completed.

- Detail Defination of test cases can begin as soon as design phase has been generated.

- Therefor all test should be planned and design before any code has been generated.

3) Pareto principle apply to software testing.

- Pareto principle implies that 80% of all errors uncover during testing will be traceable to 20% of all program components.

4) Testing should begin "in the small" and progess towards testing "in the large".

- The first test planned and executed generally focus on individual program modules.

- As testing progress, testing shift focus in attempt to find errors in integrated component and altimately testing towards entire system.

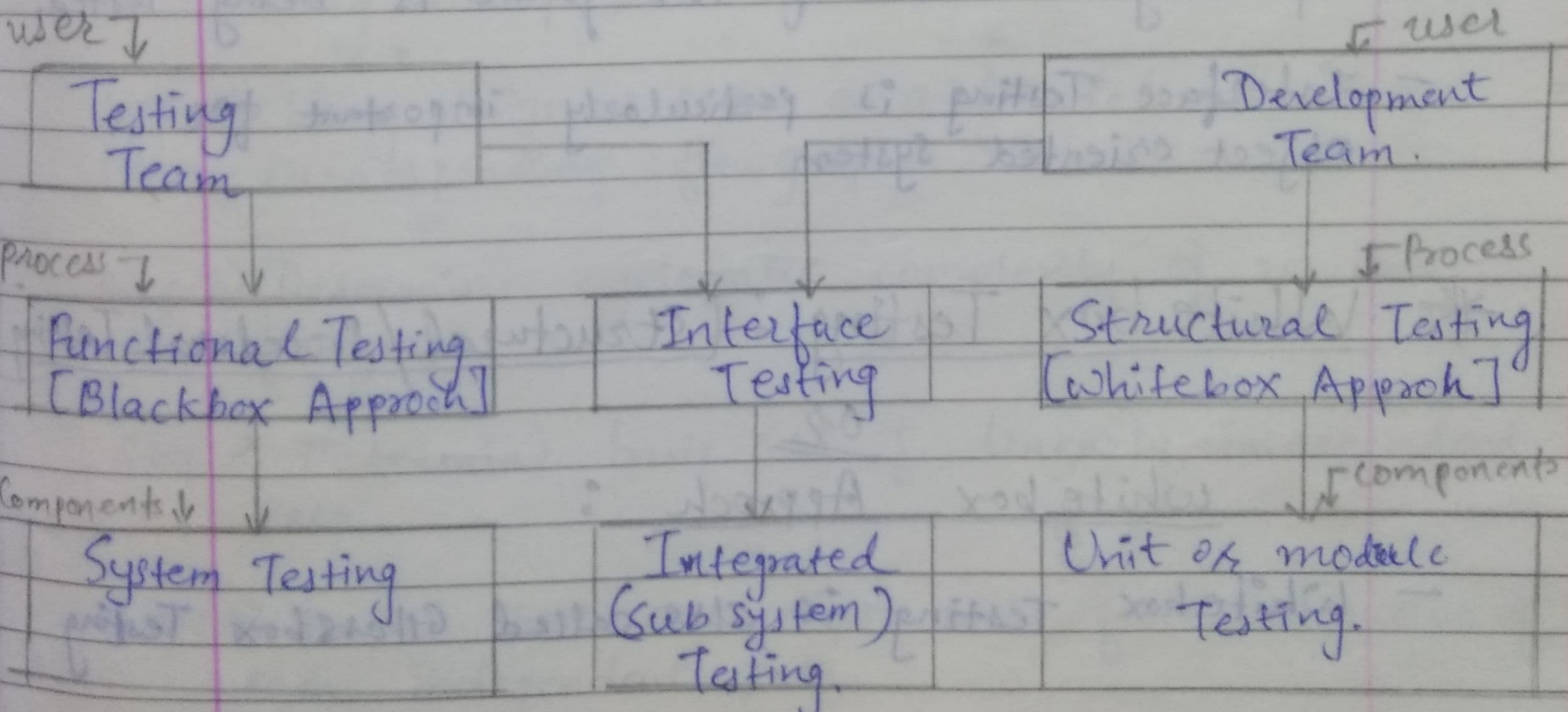
5) Exhaustive testing is not possible for any system.

- Number of path permutative for ~~customize~~ customize medium size project is very large. For this reason, It is impossible to ~~execute~~ execute every combination of path during testing.

6) To be most effective, testing should be conducted by independent 3rd party.

* Testing Process :

- The objective of Testing is process of executing the program with intent of finding errors
- Testing is intended to exercise a system so that hidden effects are exposed before system is delivered.
- There are 3 approches to detect defect testing.
  1) Functional Testing [blackbox approch]
  2) Structural Testing [white box approch]
  3) Interface Testing
- following diagram represent which approch is most suitable to which component and also indicate who should perform it.



In,

- Functional Testing, Test cases are derived from program specification.
- In, strural testing, Test cases are derived from knowledge of program Structure & Implementations.
- In Interface Testing, Test cases are derived from program specifications + knowledge of Internal Interface.
- Functional Testing is usually a responsibility of Seperate Testing Team. Unit & Module Testing [~~Strural~~ Structural Testing] are usually responsibility of development Team.
- Interface Testing may sometimes be carried out by development team or sometimes by ~~de~~ Testing team depending on nature of Sub system is being Tested.
- Interface Testing is particularly important for object oriented System

* White box Testing : [Structural / Procedural Testing]

or

White box Approch :

- Whitebox Testing is also called Glassbox Testing

- Whitebox Testing is a Testcase design philosophy that uses control structures describe as a part of component level design to derived Test cases.

- Using whitebox Testing, Software Engineer can derived Test cases that,
  1) Guarentee that all independent path within a module have been executed atleast once
  2) Test all logical decision on their True & false side.
  3) Test all loops at their boundry and within their operational bounds.
  4) Test Internal Data Structure to ensure their validity.

- following steps to derived Test cases using whitebox Testing techniques.

Step-1) Using design or code as a foundations, draw corresponding flow graph.

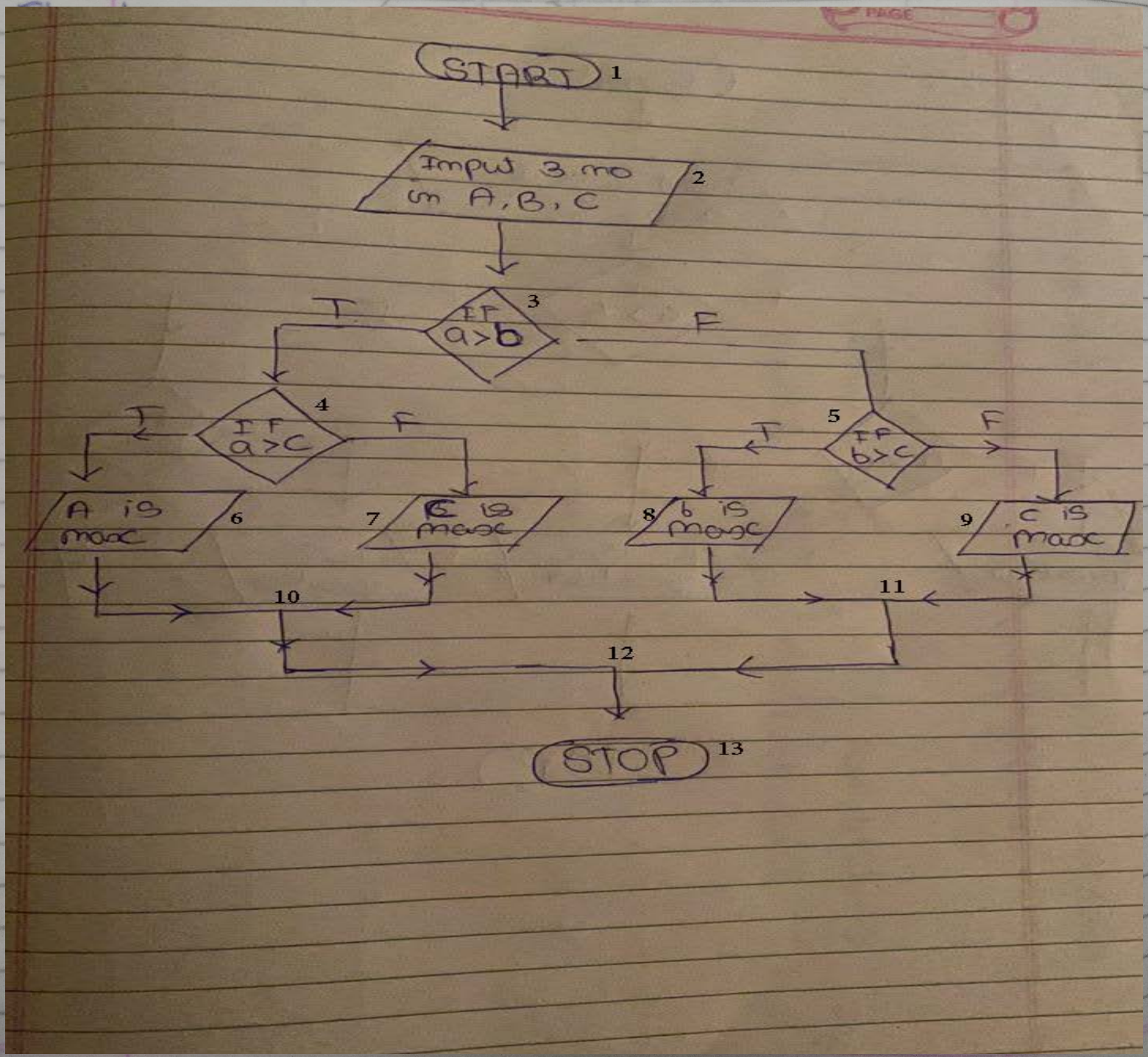
Step-2) Determine cyclomatic complexity of resultant flowgraph

Step-3) Determine basic set of linearly independent path.

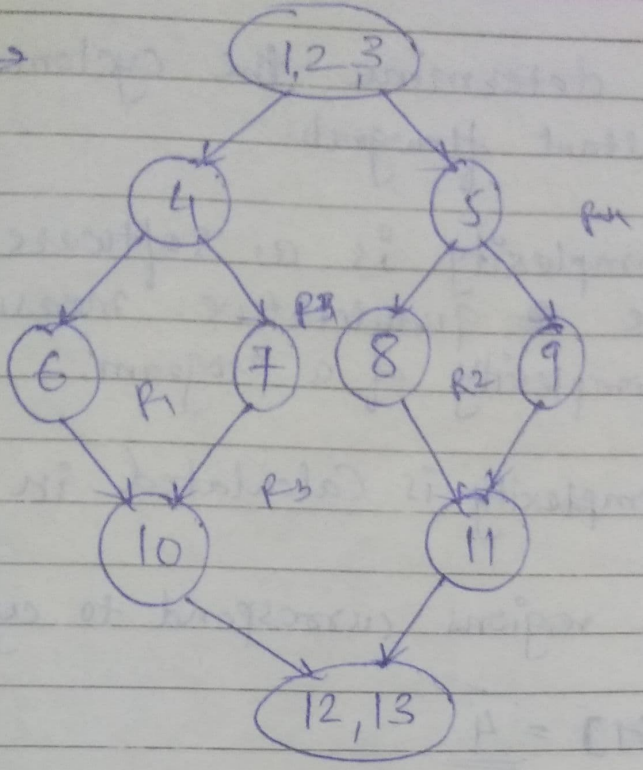
Step-4) Prepare test cases that will force execution of each independent path.

Step-1) Using code or design as a foundation, draw corresponding flowgraph.

- Flowgraph depicts logical control flow using notations for representation of control flow.



Flowgraph →



- Each circle represent 1 or more statement called flowgraph node.
- Here, N = 10 no. of node.
- The arrow on flowgraph called edges. represent flow of control.
- Here E = 12 no. of edges.
- A node that contain condition is called predicate node.
- Here P = 3 no. of predicat.
- Area are bounded by edges and nodes called region.
- Here R = 4 no. of region.

Step-2) To determine the cyclomatic complexity of resultant flowgraph

- cyclomatic complexity is a software matric that provide a quantitative measure of logical complexity of a program.
- Cyclomatic complexity is calculated in one of the 3 way:

1) The no. of regions correspond to cyclomatic complexity

- Here $R = V(G1) = 4$

2) Cyclomatic complexity is define as $V(G1) = E - N + 2$
$= 12 - 10 + 2$
$= 4$

3) Cyclomatic complexity is define as $V(G) = P + 1$
$= 3 + 1$
$= 4$

Step-3) Determine basis set of linearly independent path.

- Independent path is any path through the program that introduce atleast one new set of processing statement or new conditions and one edge has not been traversed repeated.

- Following independent path are derived from resultant flowgraph as follow.

1) 1 2 3 - 4 - 6 - 10 - 12-13
   1 2 3 - 4 - 7 - 10 - 12-13
   1 2 3 - 5 - 8 - 11 - 12-13
   1 2 3 - 5 - 9 - 11 - 12-13

STEP-4)Prepare test case that will force execution of each independent path

	Input Values			Output	
No.	A	B	C	Expected	Actual
1	10	8	7	A	
2	10	8	11	C	
3	12	15	9	B	
4	12	15	20	C	

* Blackbox Testing : or
  Functional Testing : or
  Behavioural Testing : or
  Blackbox Approch

- It focus on functional requirement of software that i.e. blackbox testing enables software engineer to derive set of input condition that will fully exersize all functional requirement for a progeam.

- So, its called functional Testing.

- In blackbox testing, Testers only know what input that can be given to the system & what output the system should give.

- So, its called behavioural Testing.

- blackbox testing is not an alternative to whitebox testing rather than it is complementory approch i.e. likely to uncover at different classes of errors than whitebox testing measures.

- Blackbox testing attempt to find error in following categories.

i) Incorrect or missing function.
ii) Interface error.
iii) errors in Datastructure or external database access.
iv) Behaviour or performance error
v) Initialization & termination error.

- Blackbox Testing is to be applied during later stages of testing. while whitebox testing early in testing process.

- following questions answer by blackbox Testing:

1) How is functional validity tested?
2) How are boundry of data classes isolated?

3) What classes of input will make good test cases!
4) What data rates & data volume can the system tolarated.!
5) What effect will specific combinations of data have on system operation.
6) Is the system particularly sensitive to certain input values.

- Applying black box testing techniqne, we derive set of test cases that satisfy following criteria.

1) Test cases that reduce by count that is greater than 1 the no. of additional test cases that must be designed to achieve reasonable Testing.

2) Test cases that tell us something about presence or absence of classes of errors rather than error associated only with test at hand.

* Interface Testing :

- It is done to check whether individual modules are communicating properly with one another as per specification.
- It is mostly used in testing user Interface of Graphical user Interface application.
- It is normally done through the use of variety of test cases.
- Interface Testing is sometimes to be carried out by developer and sometimes by testing in depend on nature of subsystem being tested.
- Interface Testing is particularly important for Object Oriented System.
- In Interface Testing, Test cases are derived from Program Specifications & knowledge of its internal interface
- In Interface Testing, Human being are able to communicate with the hardware through the Interface.
- Interface is actually a software that consist of set of messages, commands, images & other features that allow communications ~~it~~ occur between hardware device & user.

- Larger company uses Interface Testing to make sure that their customer will not encounter any problem while using particular software product, once it is delivered

- Developer usually want their product supported by more than one languages, so developers need to understand the inteface.

- During Interface Testing, program flow is checked & evaluated to determine if it mathes natural steretegy of user it navigating within the applications.

Level of Testing :

- Except for small program, System should not be tested for a single unit.

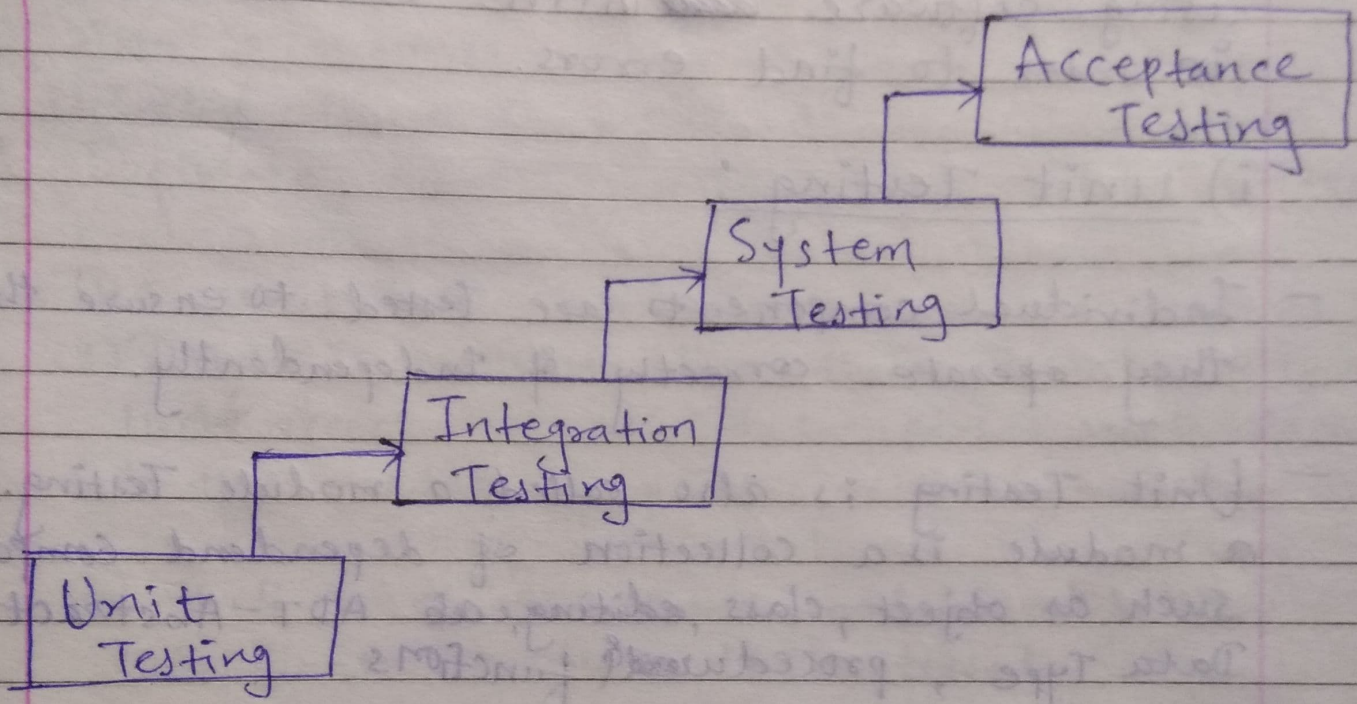
- A large System is a collection of sub system which is again the collection of modules which is again collection of function & procedure

- Therefor Testing process should proceed in levels (stages) where testing is carried out incrementally in conjuctions with system implementation

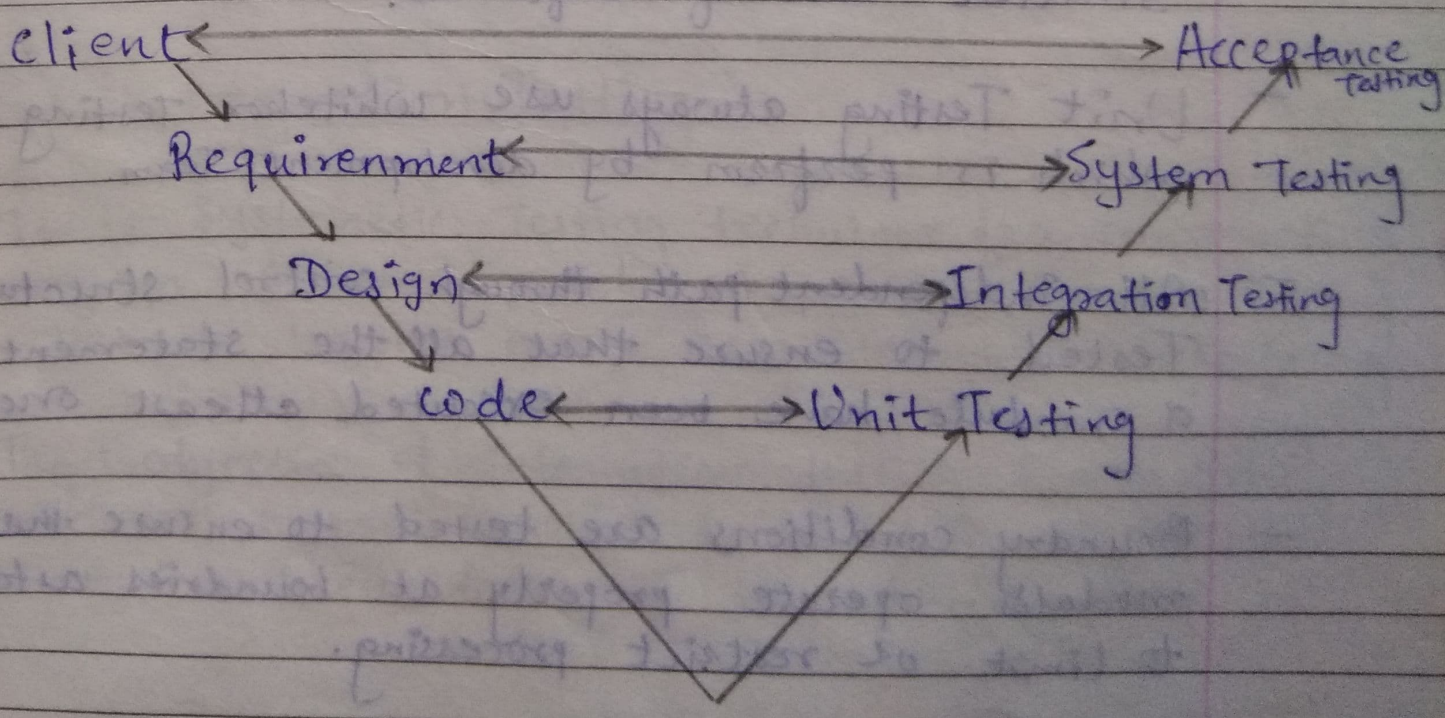
- The process is interactive one (1) with information being feedback from the next stages to the earlier

parts of process.

- The level of Testing are as follow.



* V & V technique :



* Verification :

- Verification refers to the process of executing a program in a simmulated environment while validation refers to the process of using software ~~code~~ in live environment in order to find errors.

i) Unit Testing :

- Individual components are Tested to ensure that they operate correctly & Independently.
- Unit Testing is also called a module Testing. a module is a collection of dependend components such as object, class, ~~editing, eab~~ ADT - Abstract Data Type, ~~procedure~~ & functions
- Unit Testing focus on verification efforts on smallest unit of software.
- Unit Testing always use whitebox Testing approch which is perform by development Team.
- All independent path through control structure are Tested to ensure that all the statement in a module have been executed atleast once.
- Boundry conditions are tested to ensure that modules operate properly at boundries ~~establish~~ to limit or restrict processing.

- **Local data structure is tested to ensure that data stored temporarily maintain its integrity during all the steps in functions execution.**

- **Since a module is not stand alone programme, driver and/or stub(dummy module) software must be developed for each unit test**

- The unit Test environment may be set as follow
  1) Driver is nothing more than a main program that accept Test cases data, passes such data to the module & print relavant result.

  2) Stub serve to replace modules that are subordinate module to be tested.

- Unit testing is simplifing when a module with high Cohesion is design.

ii) Integration Testing :

- It is Systemetic Testing technique for constructing program structure, while at same time conducting Test to uncover errors associated with interfacing.

- The objective of integration testing is to take unit tested module and build program structure that has been decteted by design phase.

- There are basically 2 approches to apply Integration Technique.
  1) Non incremental approch.
  2) Incremental approch.

1) Non incremental Approch [Big Bang Approch]:

- All modules have been combine in adrance & entire program structure is tested as a whole using Big Bang approch.

* Adrantage:

- set of errors are encountered.

* Disadrantage:

- correction is very difficult because of isolation of causes is complicated by large scope of program.

—— It is applicable to small project & small system.

2) Incremental Approch:

- The program is constructed & Tested in small segments where errors are easily isolated & correct, interface are to be tested completely & systemetic test approch may be applied as follow.
  A) Top down integration.
  B) Bottom up Integration.
  C) Sandwich Integration.

A) Top down Integration :

- It is incremental approch to concon construct a program in systemetic way.
- Modules are integrated by moving downword through the control Hierarchy, begining with main control module.
- Modules are incorporated into structure in depth first or breadth first manner.
- The Integration process is performed in following steps :

Step-1 : The main control module is used as a Test driver & Stub are replace for all modules directly subordinate to the main Control module.

Step-2 : Depending on Integration Stretergy selected i.e. depth first or breadth first, subordinate Stub are replace one at a time, with actual module.

Step-3 : On completion of each set of test, another stub is replaced with actual module.

Step-4 : regression testing i.e. conducting all or some of the previous cases may be conducted to ensure that new error have not be introduce.

B) Bottom-up Approch :

It begins construction & Testing with atomic modules i.e. module at lowest level in program structure.

Since modules are integrated from Bottom up, processing require for modules subordinates to a given level is always available & need of for Stub eleminations.

Bottom up Integration approch may be implimented with following steps :

Step-1 : Low level modules are combine into cluster (Grouping) & perform specific software Sub function.

Step-2 : ~~Dri~~ At driver i.e. main control program for Testing is written to co-ordinate test input & output

Step-3 : The cluster is Tested.

step-4 : Drivers are remove & cluster are combine moving upward direction in program structure.

c) Sandwich Approch :

- It consist of combination of both top down & bottom up approch.
- Sandwich Approch occure at both highest level modules & also at lowest level modules simultaneously.
- Sandwich approch may be implimented with following steps.

Step-1) High & low level modules are grouped based on control & Data processing.

Step-2) Integration within the group progress in alternating steps, between high & low level modules group.

Step-3) When integration for certain functional group is completed, integration & Testing more into next group.

iii) System Testing :

- After Integration Testing is completed, The application is deployed into logical container (real environment) then it become system.
- If one perform testing on this system then it is called system Testing.

- The most common problem arise in large Software, System & Sub System interface mismatch.
- Therefor, sub system test process should concentrate on detection of interface error by exercising this interface.
- The Testing process is concerned with validity that the system needs its functional & non-functional requirement.
- In System Testing, following performance Criteria are,
  1) Turnaround Time
  2) Backup
  3) protection
  4) personnel of candidate System.

1) Turn Around Time:

- Turn around time elapsed time between reciept of input & availability of output.
- In online System, high priority processing is handled during peak hours while law priority processing is handled later in the day or during night shift.
- The objective is to decide on an evaluate all the factors that might have handling turn

around time for the system.

2) Backup :

- It related to procedure to be used when the system is down.
- Backup plans might call for the use of another computer
- The software for candidate System must be tested for compatibility with backup computer.
- In the case of partial computer system break down, provisions must be made for the dynamic reconfiguration of the system.

3) Protection :

- File protection pertains to storing a file in seperate area for protection against fire, flood or natural disaster.
- Plan should be established for reconstructing files damaged through the hardware mal function.

4) Personnel of candidate System :

- It is also consider the facility like desk & chair, lighting, air-conditioning of place [AC] where the system is being perform & other environment

factors like noise are evaluated

4) Acceptance Testing:-

After the entire system testing is complete then acceptance testing is performed. This is the final test in the testin process before the system is accepted for operational use. It is sometimes called alpha testing. It involves testing th system with data supplied by system procures rather than simulated data developed as part of the testing process. I often reveals errors and omissions in the system definition.

When a system is to be tested as a software product, a testing process called beta testing is often used. It involves delivering a system to a number of potential customers who agree to use that system and provide feedback. Repairing the program defects may introduce new defects so testing should be repeated after the system is modified. This is sometimes called regression testing.